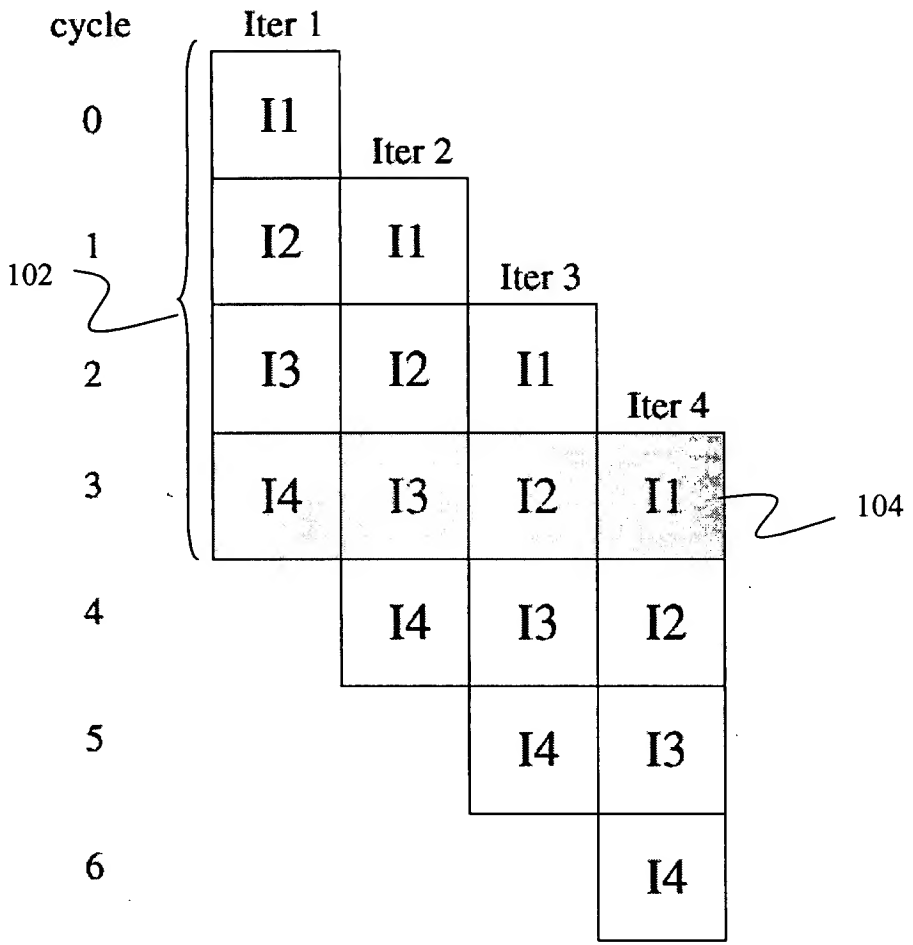

cycle
Iter 1
Iter 2
Iter 3
Iter 4
0
1
2
3
4
5
6
102
104
I1
I2
I1
I3
I2
I1
I4
I3
I2
I1
I4
I3
I2
I4
I3
I4

FIG. 1A

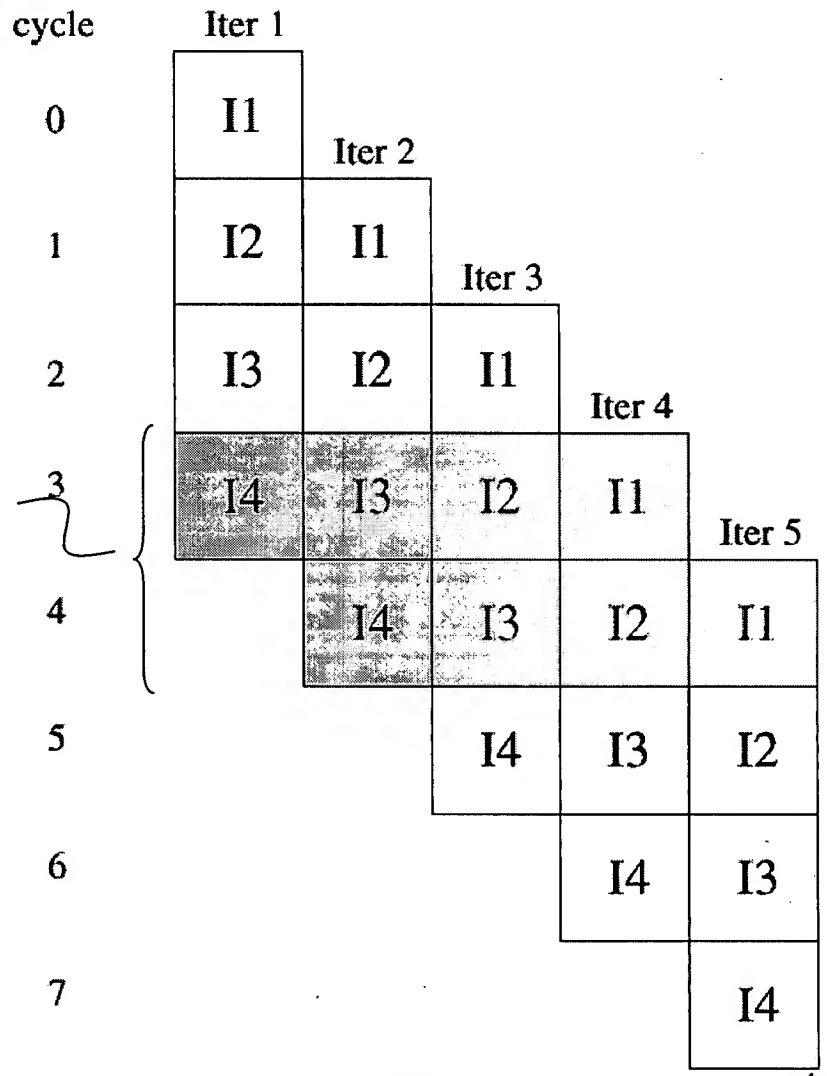

cycle
Iter 1
Iter 2
Iter 3
Iter 4
Iter 5
0
1
2
3
4
5
6
7
I1
I2
I1
I3
I2
I1
I4
I3
I2
I1
I4
I3
I2
I1
I4
I3
I2
I4
I3
I4

FIG. 1B

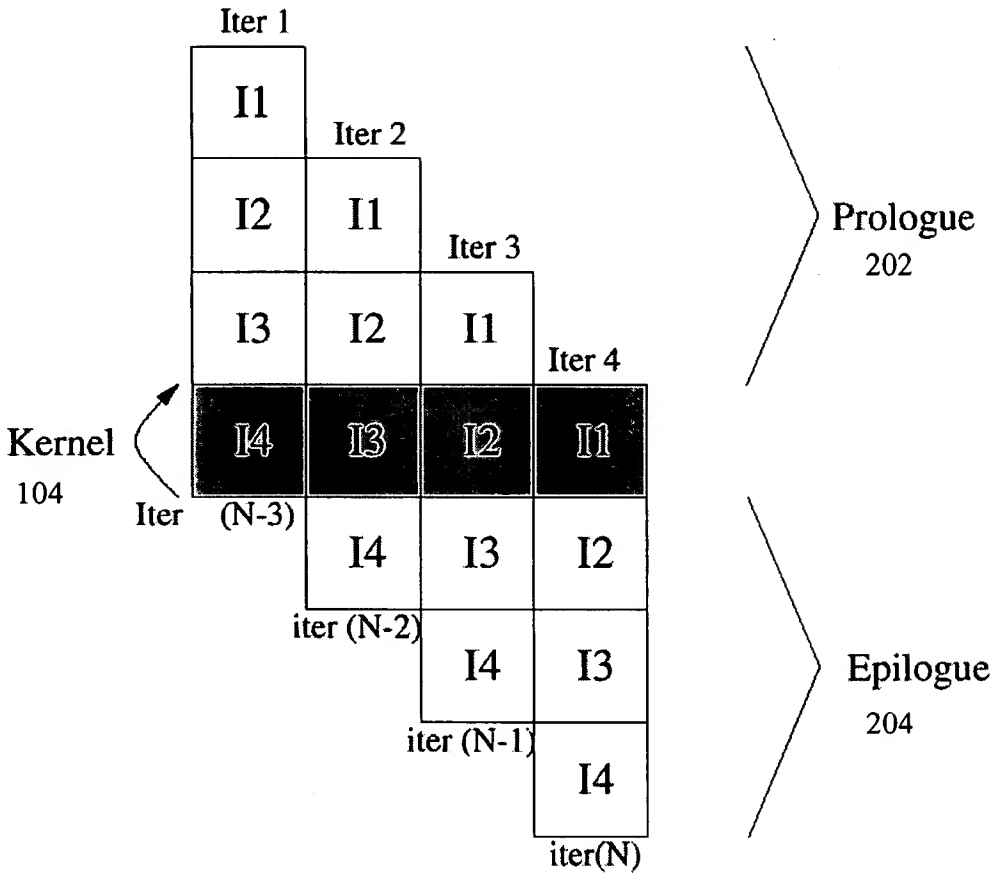

Iter 1
Iter 2
Iter 3
Iter 4
I1
I2
I3
I4
Kernel
104
Iter
(N-3)
iter (N-2)
iter (N-1)
iter(N)
Prologue
202
Epilogue
204

FIG. 2

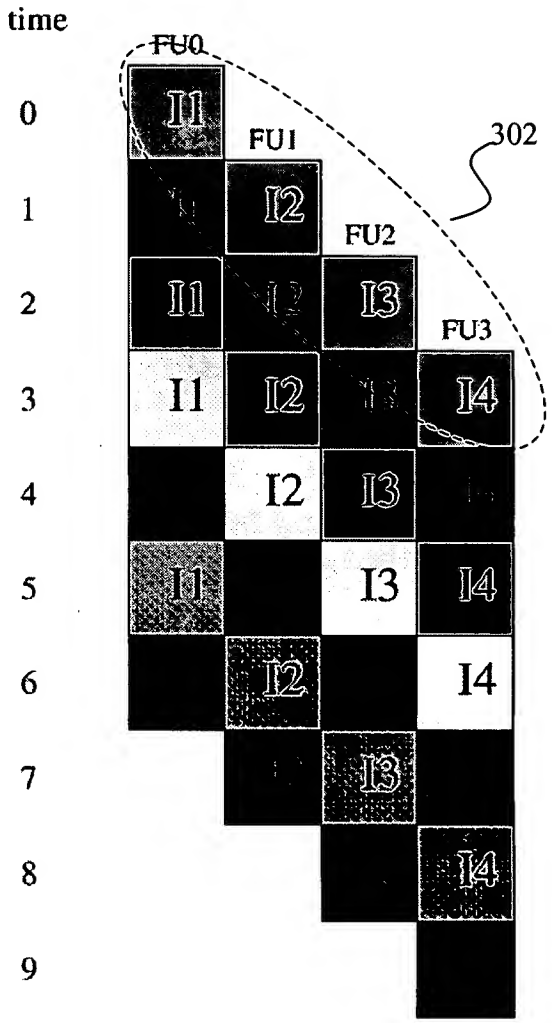

time
0
1
2
3
4
5
6
7
8
9
FU0
FU1
FU2
FU3
302
I1
I2
I3
I4

FIG. 3

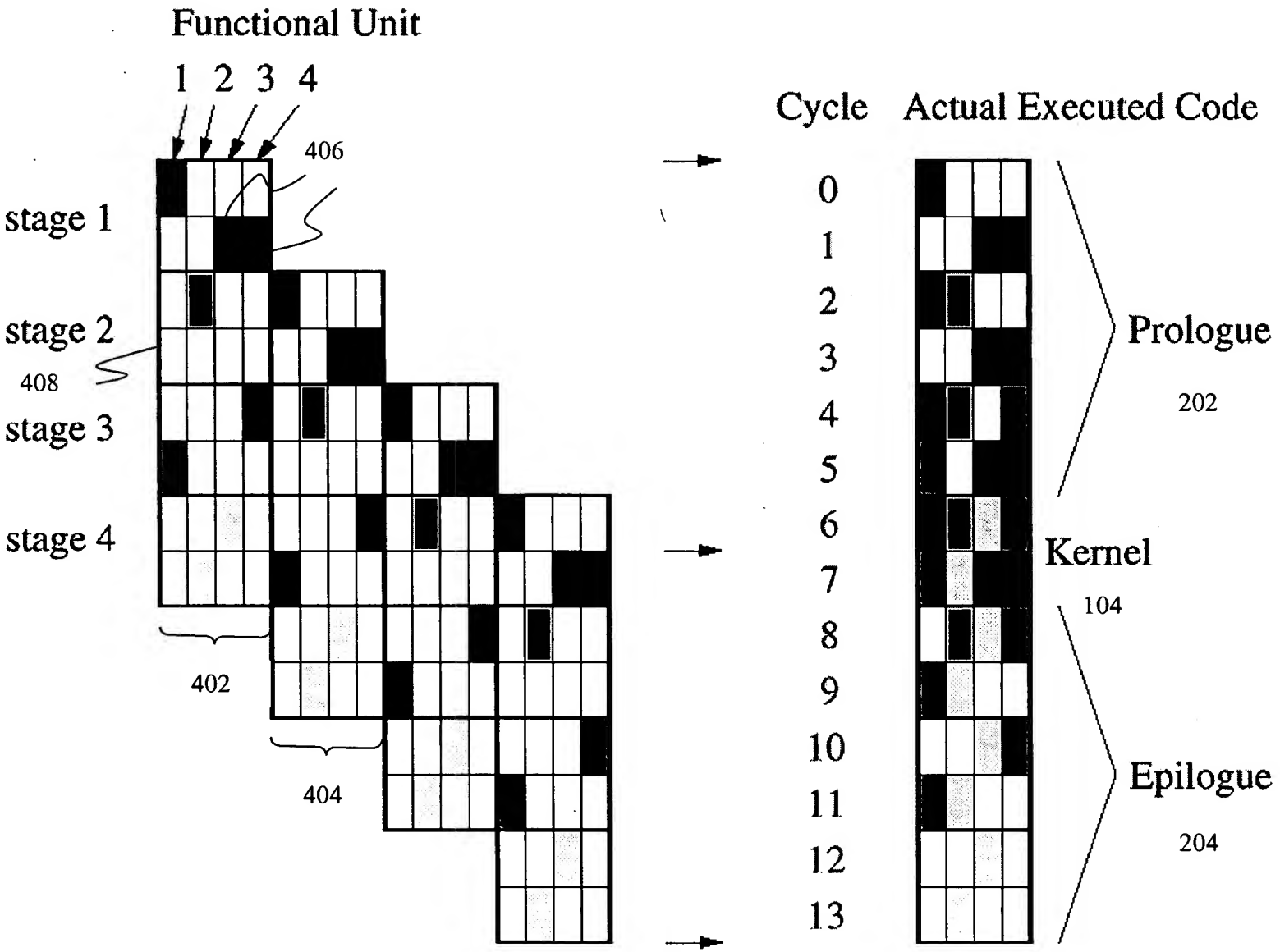

Functional Unit
1 2 3 4
406
stage 1
stage 2
408
stage 3
stage 4
402
404
Cycle
Actual Executed Code
0
1
2
3
4
5
6
7
8
9
10
11
12
13
Prologue
202
Kernel
104
Epilogue
204

FIG. 4A

FIG. 4B

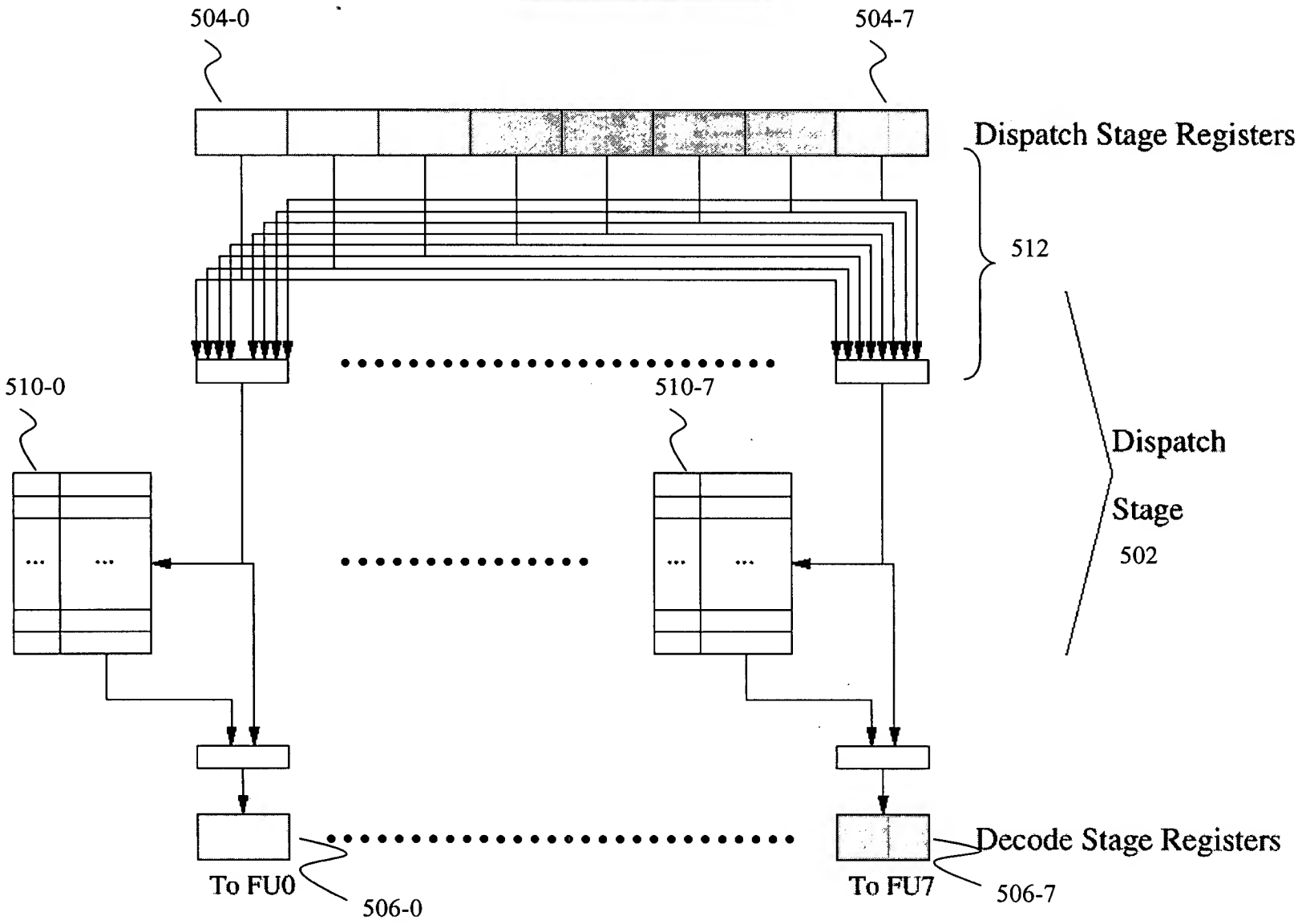

504-0
504-7
Dispatch Stage Registers
512
510-0
510-7
Dispatch
Stage
502
To FU0
506-0
To FU7
506-7
Decode Stage Registers

FIG. 5

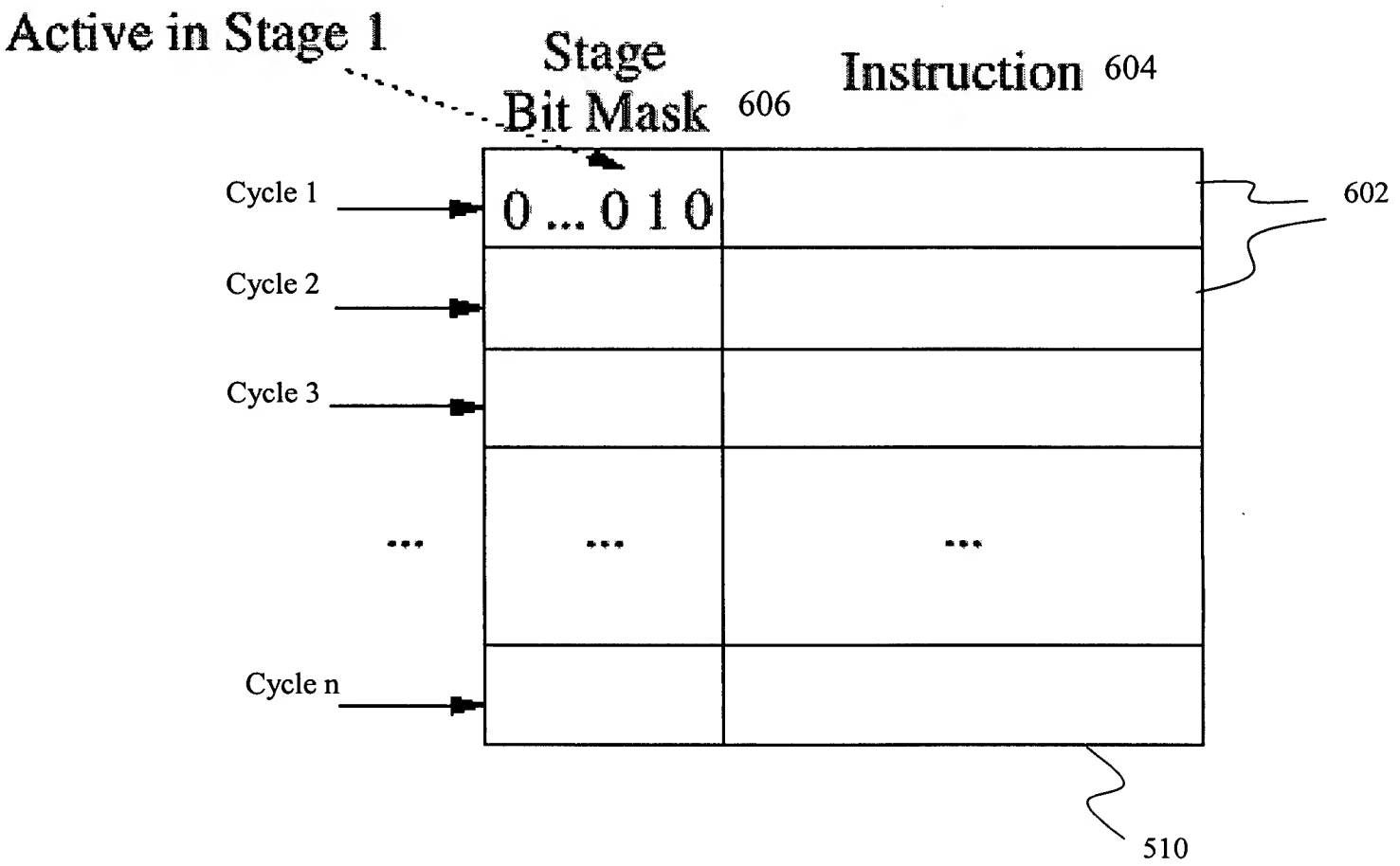

Active in Stage 1
Stage Bit Mask
606
Instruction
604
Cycle 1
0 ... 0 1 0
Cycle 2
Cycle 3
...
...
...
Cycle n
602
510

FIG. 6

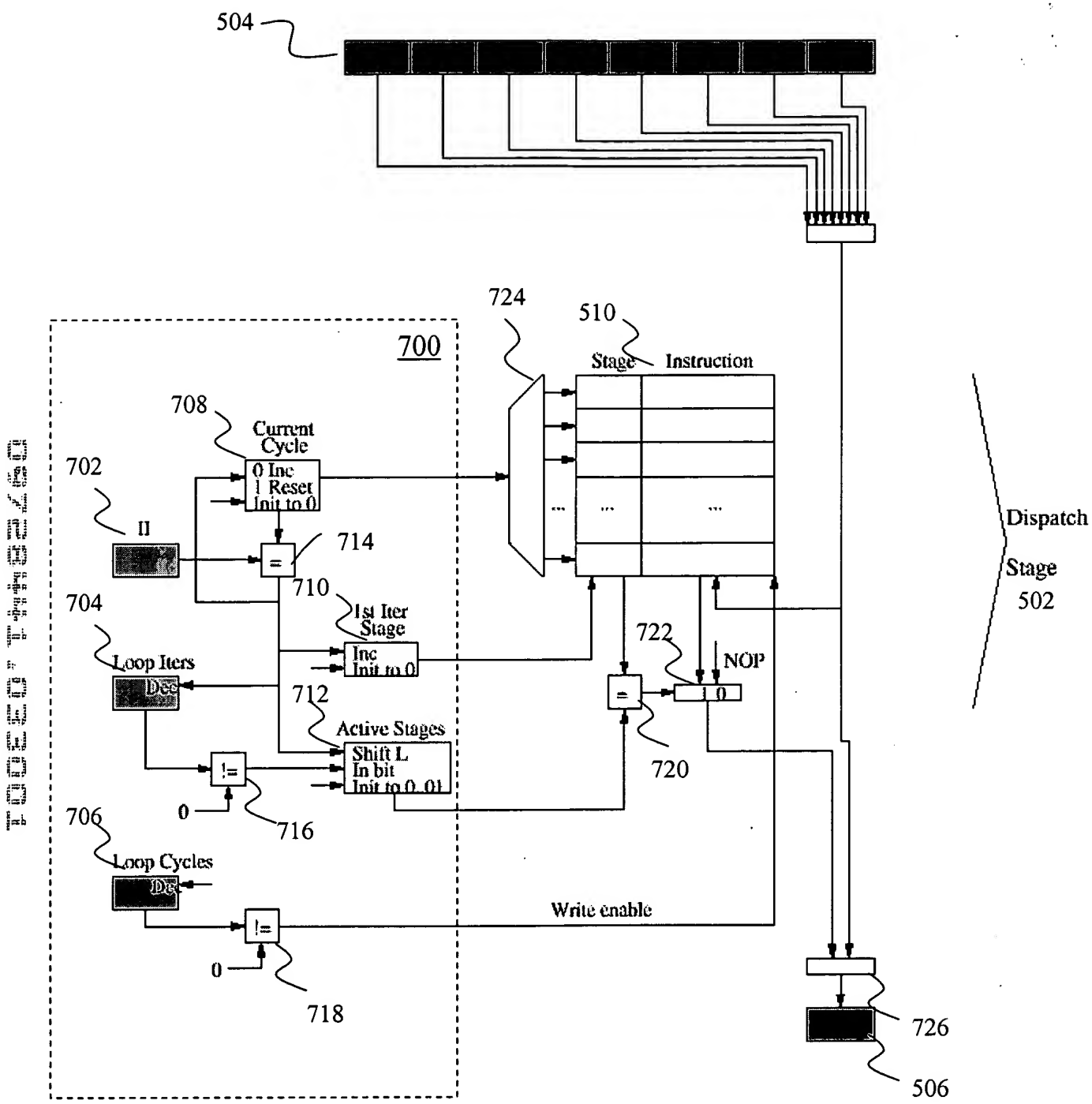

504
700
724
510
Stage
Instruction
708
Current Cycle
0 Inc
1 Reset
Init to 0
702
II
714
710
704
1st Iter Stage
Inc
Init to 0
Loop Iters
Dec
712
Active Stages
Shift L
In bit
Init to 0 01
!=
0
716
706
Loop Cycles
Dec
!=
0
718
722
NOP
1 0
=
720
Write enable
Dispatch Stage 502
726
506

FIG. 7

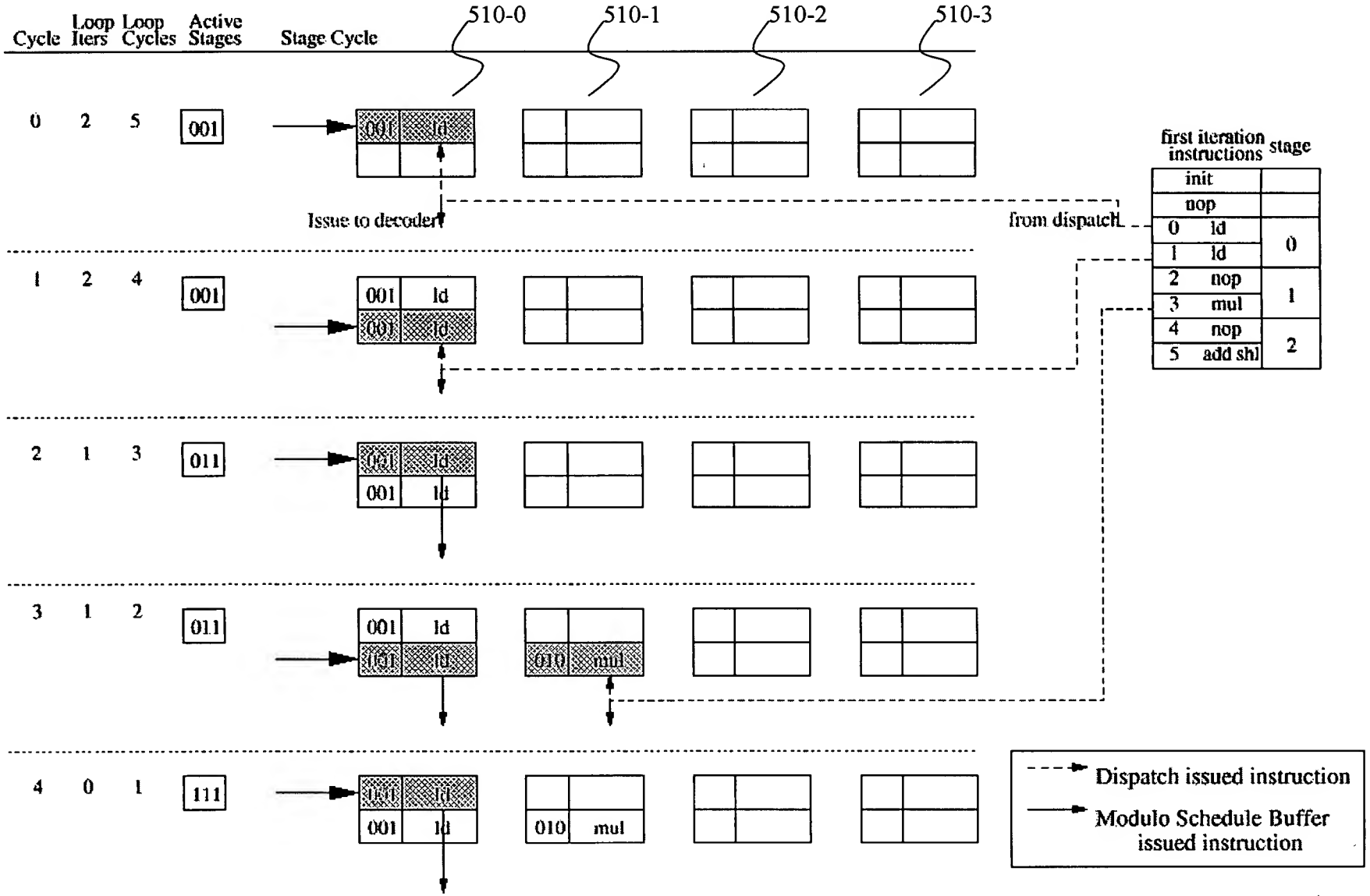

Cycle
Loop Iters
Loop Cycles
Active Stages
Stage Cycle
510-0
510-1
510-2
510-3
0 2 5 001
1 2 4 001
2 1 3 011
3 1 2 011
4 0 1 111
001 ld
010 mul
Issue to decoder
from dispatch
first iteration instructions
stage
init
nop
0 ld
1 ld
2 nop
3 mul
4 nop
5 add shl
0
1
2
Dispatch issued instruction
Modulo Schedule Buffer issued instruction

FIG. 8A

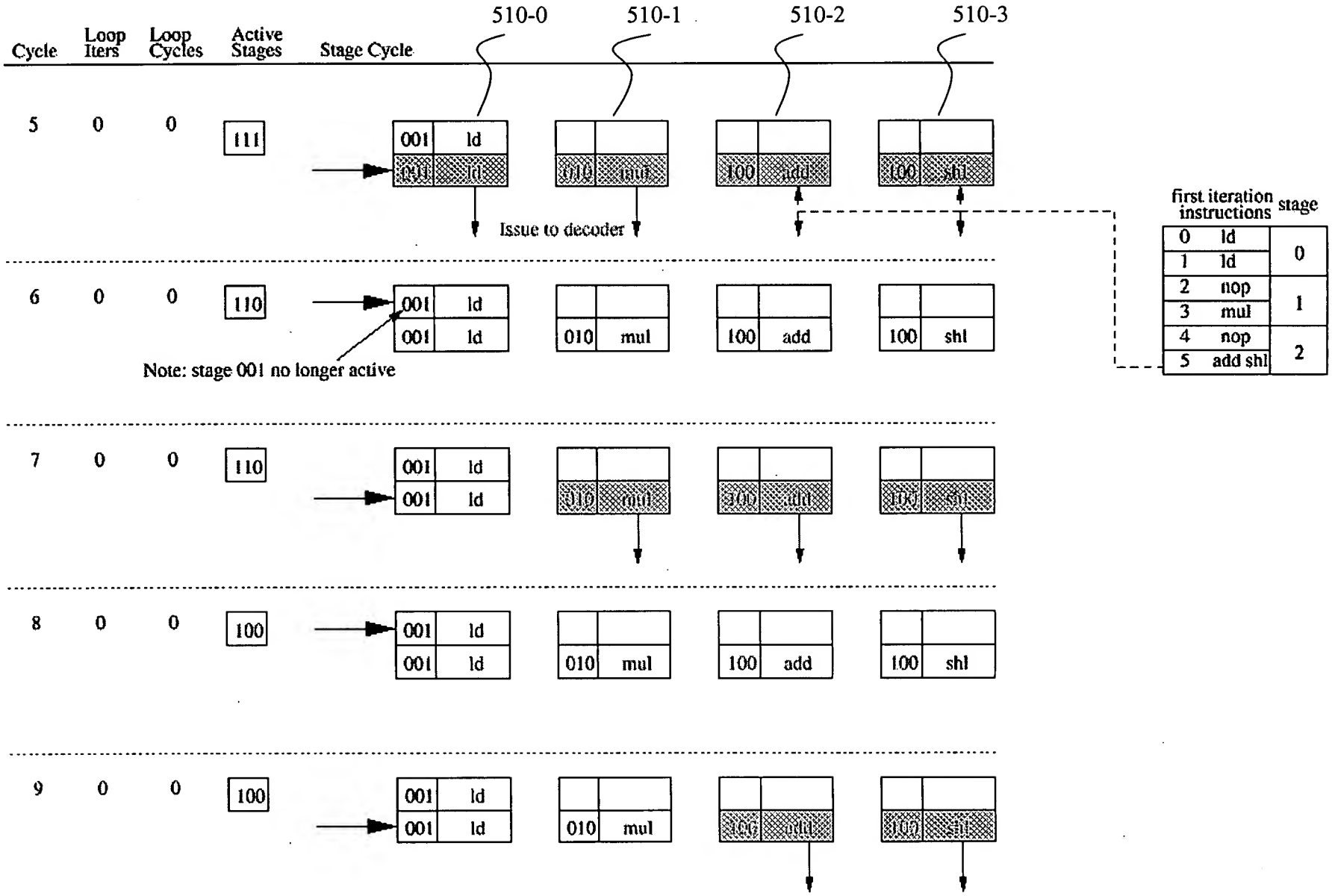

510-0
510-1
510-2
510-3
Cycle
Loop Iters
Loop Cycles
Active Stages
Stage Cycle
5 0 0 111
001 ld
010 mul
100 add
100 shl
Issue to decoder
6 0 0 110
001 ld
001 ld
010 mul
100 add
100 shl
Note: stage 001 no longer active
7 0 0 110
001 ld
001 ld
010 mul
100 add
100 shl
8 0 0 100
001 ld
001 ld
010 mul
100 add
100 shl
9 0 0 100
001 ld
001 ld
010 mul
100 add
100 shl
first iteration instructions
stage
0 ld
1 ld
0
2 nop
3 mul
1
4 nop
5 add shl
2

FIG 8B

cycle	RTL
0	r1 <- 0
1	r1 <- r1 * 5
2	r1 <- r1 + 4
3	st [r2], r1
4	st [r3], r1

inst.	latency
mul	3
add	1

FIG. 9

EQ model or
LE model w/ actual
MUL latency of 3

cycle	RTL	
0	r1 <- 1	
1	r1 <- r1 * 5	
2	r1 <- r1 + 8	
3	st [r2], r1	; store 9 into [r2]
4	st [r3], r1	; store 5 into [r3]

FIG. 10A

EQ model w/ int. after cycle 2 or
LE model w/ actual
MUL latency of 2

cycle	RTL	
0	r1 <- 1	
1	r1 <- r1 * 5	
2	r1 <- r1 + 8	
3	st [r2], r1	; store 5 or 9 into [r2]
4	st [r3], r1	; store 5 or 9 into [r3]

FIG. 10B

EQ model w/ int. after cycle 1 or
LE model w/ actual
MUL latency of 1

cycle	RTL	
0	r1 <- 1	
1	r1 <- r1 * 5	
2	r1 <- r1 + 8	
3	st [r2], r1	; store 13 into [r2]
4	st [r3], r1	; store 13 into [r3]

FIG. 10C

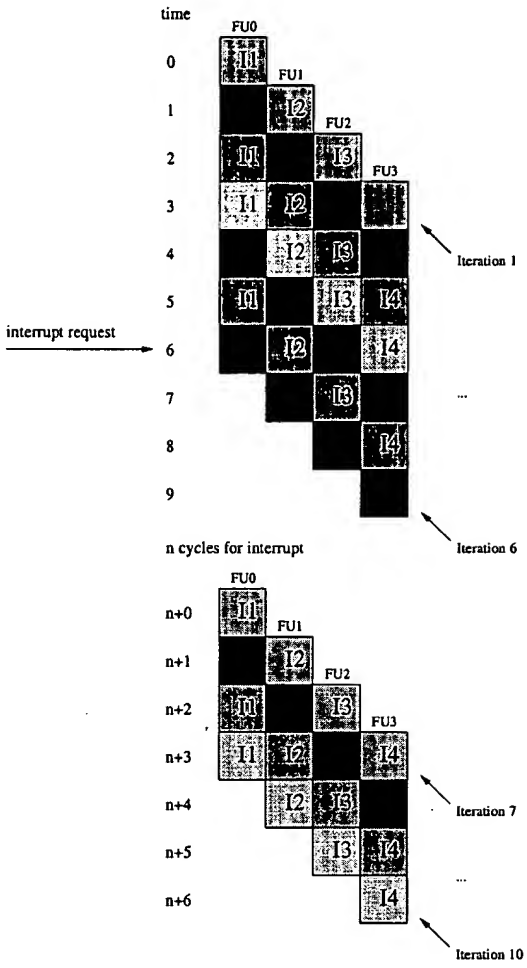
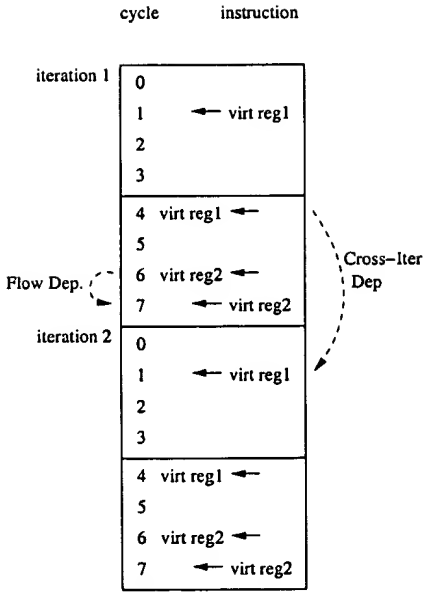
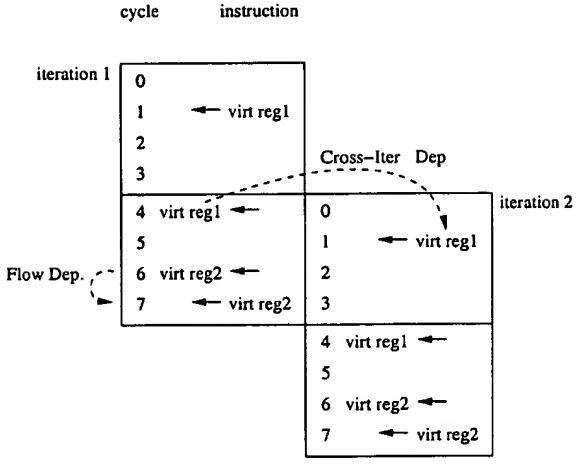


FIG. 11



(a) Traditional register allocation of the loop body.

FIG. 12A



(b) Modulo schedule-aware register allocation of the loop body.

FIG. 12B

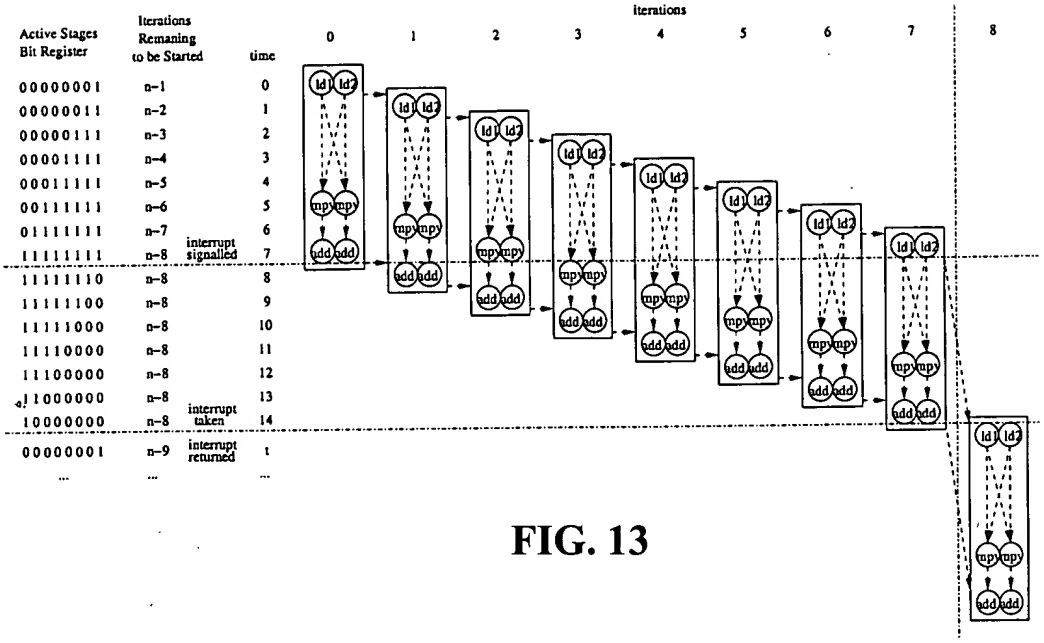


FIG. 13

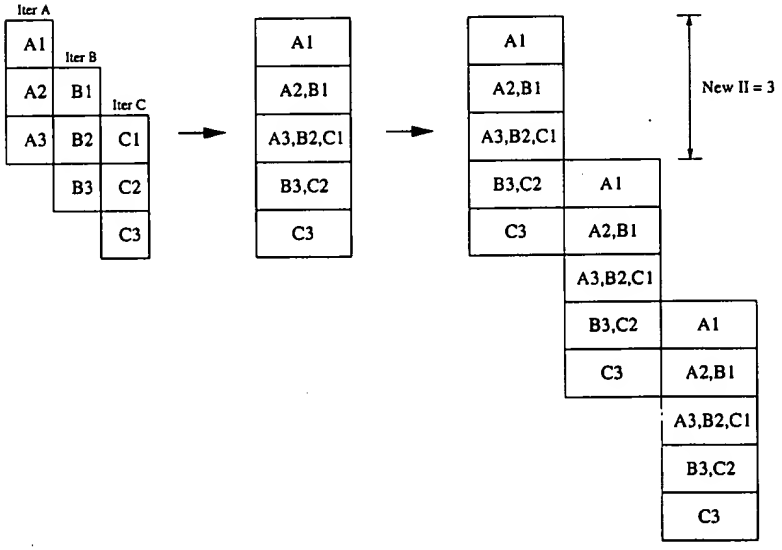


FIG. 14A FIG. 14B FIG. 14C